

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

METHOD AND APPARATUS FOR
MULTIPLE MEDIA DIGITAL COMMUNICATION SYSTEM

Field of the invention

The present invention relates to the field of digital communications systems, and more particularly to systems transporting multiple media (multimedia) and/or communicating such multimedia through a plurality of connections to multiple callers.

Background of the invention

In the prior art, multimedia communications, such as videoconferencing systems for providing two way video and audio, are well known. Given sufficient bandwidth and dedicated independent channels, (e.g. 6 Mhz for an analog video channel, 3 Khz for an audio link over a standard analog telephone line, etc), videoconferencing between two callers can be realized. However, communication channels providing 6 Mhz video bandwidth are not generally or universally available. A major obstacle to wide spread implementation and acceptance of multiple media conferencing systems is the limited bandwidth of the available communication channels. In addition, typical communication channels available on packet switched networks such as AppleTalk, from Apple Computer,

California, USA, or Netware from Novell Inc, Oregon, USA, do not provide the continuous real time analog or digital connection of a telephone line or modem. Instead, packet switched networks provide non-real time bursts of data in the form of a switched packet containing a burst of digital data. Thus, in addition to bandwidth limitations, packet switched networks present delay limitations in implementing real time multiple media conferencing systems. The same bandwidth and time delay limitations which apply to all time division multiple access (TDMA) communication systems and similar schemes present obstacles to achieving real time multimedia communications.

Typically, the problem of videoconferencing two callers is approached by compressing the composite video signal so that the resulting transmitted data rate is compatible with the available communication channel, while permitting acceptable video and audio to be received at the other end of the communication channel. However, solutions in the past using lossy compression techniques, have been limited to compromising quality in order to obtain acceptable speed. Recently, non-lossy compression techniques have become available. The problem still remains as to how to match the bandwidth and timing constraints of available digital formats to the available communication channels, both present and future.

Summary of the invention

The present invention is embodied in a digital communication system where multiple media data sources are time multiplexed into a packetized data stream. At both the transmit side, and the receive side, audio packets are given priority processing over video packets, which in turn have priority over text/graphics data packets. Continuous real time audio playback is maintained at the receiver by delaying the playback of received audio in a first in/first out (FIFO) buffer providing a delay at least equal to the predicted average packet delay for the communication system. Optionally, the average system delay is continuously monitored, and the audio and video playback delay time as well as audio and video qualities are adjusted accordingly. In another embodiment of the invention, a conference of three or more callers is created by broadcasting a common packetized data stream to all conference callers. Use of the present invention further permits an all software implementation of a multimedia system.

1. In accordance with a first aspect of the present invention, multiple data sources forming data packets are combined into a prioritized data stream.

The present invention is embodied in a method and apparatus for combining data from a plurality of media sources into a composite data stream capable of supporting simultaneous transmission including multiple video and graphic signals and real time audio. Video, audio and other signals are integrated in a

non-standard transmission format determined by a novel streaming algorithm and prioritization scheme designed to provide the best balance between transmission quality and realization of real time rendition of each.

For example, each data type packet at the transmitter is assigned a priority between 0 and 10000, with 0 being the highest priority and 10000 the lowest. An audio packet is given priority 20, a video packet is given priority 50. Screen data packets and file data transfer packets are both given priority 180.

Before transmission on the communication channel, packets are placed in a queue according to priority order. As new packets are generated, the queue is reorganized so that the new packet is placed into its proper priority order.

At the receiver, each task runs according to its assigned priority. Packets with priorities between 0 and 100 are processed first, to the exclusion of packets with priorities 101 through 10000. Audio, being the highest priority (20), is processed first to the exclusion of all other packets. Within the class of packets with priorities between 101 and 10000, packets are processed according to relative priority. That is, higher priority tasks do not completely shut out tasks of lower priority. The relationship among priorities is that a priority 200 task runs half as often as a priority 100 task. Conversely, a priority 100 task runs twice as

often as priority 200 task. Tasks with priorities between 0 and 100 always run until completion. Thus, video, screen data and file data, processing tasks are completed after audio processing in accordance with the relative priority of the packets.

A multi-tasking executive dynamically reassigns task priorities, to efficiently complete all tasks within the available time, while performing the highest priority tasks first. At any given time, there are different tasks all at different priorities, all yielding to each other. In general, a task yields to a higher priority task, if it is not running an uninterruptable sequence. If the current task completes its cycle, its priority is reassigned to a lower priority. If the priority of two or more tasks is equal, then the multi-tasking executive executes each task in a round robin fashion, performing a portion of each task, until the completion of all tasks with the same priority.

The assignment of packet priorities, and processing according to priority assures that audio will be given precedent over video, while audio and video will be given precedent over both screen data and file transfer data.

As indicated above, continuous real time audio playback is maintained at the receiver by delaying the playback of received audio in a first in/first out (FIFO) buffer having a size at least equal to the predicted average packet delay for the communication

system. Optionally, the delay of the audio FIFO may be made variable. A variable delay audio FIFO buffer at the receiver allows the system to shrink or grow the time delay between one machine and the other. The ability to shrink or grow the difference in time between the sender and receiver permits the system of the present invention to compensate for indeterminate system delays. If the changes are slight, the difference in pitch is not noticeable. For greater changes, the technique of audio resampling may be used to increase or decrease the rate of audio playback without changing the pitch of audio content.

Similarly, video playback continuity at the receiver may also be improved by delaying the playback of received video in a first in/first out (FIFO) buffer having a size at least equal to the predicted average packet delay for the communication system. The delay of the video FIFO may be made variable, allowing the system to shrink or grow the time delay between one machine and the other to compensate for indeterminate system delays. Again, if the changes are slight, the change in frame rate is not noticeable. However, video data does not age as quickly as audio data. Therefore a smaller video FIFO can be used. Also, a video image may have short discontinuities without a perceived loss of the video connection. Audio playback, on the other hand, is more sensitive to discontinuities, and it is more important to maintain continuity at the receiver. Ideally, when both audio and video are used in a multimedia conference, the delay for audio and video

should be equal to make sure that they are synchronized. In the latter case, the actual system delay is calculated by finding the maximum delay of both audio and video packets.

Data from media sources tend to come in bursts. For example, audio data rates rise when speaking, and fall to zero during a silence. In the present embodiment, the silence between words provides the present system with an opportunity to catch up by refilling the audio FIFO buffer before it empties. In such manner, the present system compensates for the delay inherent in a packet switched, time delay variant, communication channel.

Similarly, video sources including graphic screen data, are generated in bursts. That is, the data rate for video ideally falls to zero when there is no motion. The data rate for transmitting screen graphics falls to zero when there are no changes. When the caller changes the screen, (such as the collaborative work document displayed on the screen), data is generated.

Thus, following the priority scheme of the present invention, video is updated only when no speech data is being processed. However, processing of speech data does not include the playing of sound. Once the sound starts playing, there is no need to further spend time to process the sound. Sound playing needs no supervision. Therefore, video updating occurs while sound is playing. After speech is playing close to real time (with a

delay), video text and graphics are updated in the background. Video, text, graphics and data files are updated at lesser priorities. Except for audio and video data, task priorities are re-assigned to assure that all tasks will be completed, and that a higher priority task will not completely prevent the lower priority tasks from being completed.

-2. In accordance with a second aspect of the present invention, multiple signal packets are broadcast to a plurality of callers to create a common multimedia conference.

In addition to assigned priorities, data packets having multiple destination addresses are broadcast over a plurality of connections to multiple callers. Each caller receives the same data packets with assigned priorities, and processes the received packets in a similar manner. As new data is generated from each caller in the video conference, new data packets are broadcast to the other callers. Thus, due to the broadcast of data packets representing audio, video and screen data, all callers are conferenced together, each seeing and hearing each other, while discussing the same screen document. Additional callers can be added to the conference over a plurality of connections without adding undue burden, because in a conference, each caller needs to generate data only once, which is then transmitted either simultaneously or sequentially depending on the kind of connection, to other callers.

3. In accordance with a third aspect of the present invention data received on a first communication medium (for example on a broadband local area network, such as ethernet) are re-broadcast on a different communication medium (such as a telephone line) in order to conference callers on the different communication media in a common multimedia conference. The present invention thereby provides the option of desktop videoconferencing on standard computer networks and telephone lines.

Brief description of the drawings

Figure 1A is a dialog box indicating connection status as it would appear on the screen of a Macintosh computer used in conjunction with the present invention.

Figure 1B is a dialog box indicating an incoming call as it would appear on the screen of a Macintosh computer used in conjunction with the present invention.

Figure 1C is a dialog box illustrating the connection status message that would appear for a call that was not accepted as it would appear on the screen of a Macintosh computer used in conjunction with the present invention.

Figure 1D is a window containing a moving video image as it would appear on the screen of a Macintosh computer used in

conjunction with the present invention.

Figure 2 is a video screen illustrating a video conference between two callers and sharing a common document.

Figure 3 is a video screen illustrating a video conference between three callers and sharing a common document.

Figure 4 is a block diagram illustrating the sequence of operations for establishing a connection in a multiple media digital communication system embodying the present invention.

Figure 5 is a block diagram illustrating the sequence of operations for establishing media types to be used in a conference call in a multiple media digital communication system embodying the present invention.

Figure 6 is an illustration of a packet data format used in conjunction with the present invention.

Figure 7A is a block diagram of a multiple media communication system transmitter in accordance with the present invention.

Figure 7B is a block diagram of a multiple media communication system receiver in accordance with the present invention.

Figure 8 is a block diagram of a method and apparatus for processing a data packet in accordance with the present invention.

Figure 9 is a block diagram illustrating the sequence of operation of a method and apparatus for processing data packets in accordance with the present invention.

Figure 10 is a block diagram of a method and apparatus for establishing a connection for specific media between callers used in conjunction with the present invention.

Figure 11 is a block diagram illustrating the sequence of data packet flows with optional acknowledgement handshake packets.

Figure 12 is a block diagram of a multiple media digital communications system for sending and receiving multiple media for a first caller in accordance with the present invention.

Figure 13 is a block diagram of a multiple media digital communications system for receiving and sending multiple media for a second caller in accordance with the present invention.

Figure 14 is a first configuration of the present invention for conducting a standard videoconference call over an ethernet network.

Figure 15 is an alternate configuration of the present invention for conducting a standard videoconference call with collaborative data over an ethernet network.

Figure 16 is an alternate configuration of the present invention for leaving a recorded multimedia message of a videoconference call with collaborative data over an ethernet network.

Figure 17 is a three caller multimedia conference call in a system in accordance with the present invention.

Figure 18 is an alternate embodiment of a three caller multimedia conference call in a system using both ethernet and a telephone modem in accordance with the present invention.

Detailed description

From the viewpoint of the caller, the present multimedia communication system operates as follows:

A caller on a desktop computer initiates a multimedia call by selecting a media type and desired connection with a second party. A dialog box of the type shown in Figure 1A appears on the screen, illustrating the connection status. Caller 2, who receives the call, views a dialog box on his screen of the type illustrated in

Figure 1B to announce an arriving call. Caller 2 has the option of deciding to either pick up or deny the call, or to take a message. If the call is denied by caller 2, then caller 1 views a dialog box 14 as illustrated in Figure 1C. For purposes of illustration, a video box 16 containing a video of the first caller 1 is shown in Figure 1D. If the caller decides to take a message, caller 2 can now control the connection and optionally send an announcement message requesting a message.

Figure 2 illustrates the screen appearance of a typical multimedia call with a collaborative document. On the screen 20 of caller 1, a video box 24 appears in which a moving video showing caller 2 appears. The screen of caller 2 is similar, but contains the image and sound of caller 1. On both the screens of callers 1 and 2 can be a collaborative document 22. Caller 1 and caller 2 are connected by moving video and two way audio and can discuss collaborative document. Caller 1 may manipulate the document and the changes will appear on the screen of caller 2. In an alternate embodiment, caller 2 may manipulate the document as well.

Figure 3 illustrates the screen 20 appearance of a three party video conference as it appears to caller 1. Caller 3 appears in a video box 26 as well as caller 2 in another video box 26, and the collaborative document 22. The other callers 2 and 3 see the other two members of their video conference on their screen as well as collaborative document 22. The size and position of video boxes 24

and 26 is selectable by caller 1. A video image 25 of caller 1 may also appear on the screen of caller 1 to let caller 1 see what is being transmitted. Reducing the size of video box 25 reduces the amount of (video) data which must be transmitted by the communication system.

CONNECTION ESTABLISHMENT

The sequence of operation for establishing a connection between caller 1 and caller 2 over a communication network is illustrated in figure 4. The network may typically be AppleTalk, ethernet or any other commonly available local area network. Also, a connection can be established over a telephone line or other proprietary digital telephone lines such as ISDN.

The terms, "connection stream" and "media stream" used below are further defined in the description of figures 8-11. For present purposes, such terms may be regarded as routines for handling data packets. Caller 1 selects a media type stream 28, and a connection stream 30, suitable for the communication medium. An initial message termed a hello packet is sent to a caller 2 connection stream 32. The connection stream 32 provides a dialog box to caller 2 informing that there is an incoming call from caller 1. Caller 2 can then decide 34 to either accept or deny the call, or alternatively to take a message if caller 2 is not present. The accept, deny and take commands are communicated back to the connection stream 32 which sends a return message across the

communication system back to connection stream 30 and caller 1.

In addition to denying or taking the call, caller 2 has the option to pick a different media type. That is, for example, if the media type stream 28 of caller 1 is video, and caller 2 does not want to accept a video call but will accept an audio call, then the return message pick will indicate that caller 2 is picking audio as the media for an incoming call. At caller 1, connection stream 30 distributes the response from caller 2. Specifically, if the call is denied then the connection attempt is deleted 40. If a different media is picked, then a modification of the media type stream 28 is performed. If take a message was selected, then the appropriate file transfer 38 takes place to transmit an announcement file, and a message is requested to be sent back.

Figure 5 illustrates the communication sequence for selecting the among the various media types between caller 1 and caller 2. For media type stream 42, a request is sent through connection stream 44 across the communication channel to caller 2 at connection stream 46 which is forwarded to media type stream 48. Caller 2 configures itself to accept the media type stream which is being presented to it by modification of its registration server 50 which clones the desired media type. If accepted, media type stream 48 sends a message through connection stream 46 across the communication medium to caller 1. The acceptance is received at connection stream 44 and communicated to media type stream 42 which

opens up the connection for the given media type between caller 1 and caller 2.

DATA FORMAT IN PACKETS WITH PRIORITY AND MULTIPLE DESTINATIONS

Figure 6 shows a packet data format 52 suitable for use in conjunction with the present invention. The length of the packet is indicated by data field 54. The length of the header is indicated by data field 56. The packet type and identification are respectively indicated by data fields 58 and 60.

The priority of the data packet is indicated in data field 62. When transporting multiple media digital data packets, the priority data field determines which of the packets has the highest priority in processing. Data fields 64 and 66 respectively contain information as to the message state, and a checksum for determining message errors. The packet source address is indicated at data field 68, and a destination count as to the number of destinations this packet will reach is indicated at data field 70. Also, an active destination count, the number of destination which have not yet received this packet, and a maximum destination count is provided in data fields 72 and 74 respectively.

The data packet 52 of Figure 6 contains a number of destination addresses 76. The plural destination addresses provides a broadcast capability by which all the callers in a

conference call can view common documents and see and hear each other. That is, when a data packet 52 contains audio data representing one speaker's voice, that packet is broadcast to all the destinations simultaneously. The same is true of the video and document updates. The destination addresses is followed by the actual variable length data of the data packet in field 78.

SYSTEM OVERVIEW

A block diagram of a multiple media communication system transmitter is shown in figure 7A. A packet with priority 20 is generated 708 from audio data source 702. A packet with priority 50 is generated 710 from video data source 704. A packet with priority 180 is generated 712 from text/graphics data source 706. A WriteQueue 716 (a buffer memory for storing packets to be sent) is provided for holding packets to be transmitted over the communication channel 718. Control 714, responsive to packet generation, 708, 710, 712 places the packets in the WriteQueue 716 in order of packet priority. In hardware, a queue maybe a FIFO. In software, WriteQueue 716 is a linked list of packet containers with pointers to the next and previous packet containers. Since the Writequeue 716 is an ordered list, adding a new entry is achieved by modifying two pointers to add the new entry to the list in the proper order.

A block diagram of a multiple media communication system

receiver is shown in figure 7B. Two substantially similar receivers, one for caller 2 and another for caller 3 are illustrated. Both callers are respectively connected to a broadcast communication channel 720, 746. A ReadQueue 722, 724 (a buffer memory for storing packets) receives packets for processing. A control means 726, 728 selects packets to be processed based on the packet priority. A multi-tasking control 730, 738 processes data packets in accordance with assigned priorities. As indicated, audio packets have the highest priority and are processed first. Other packets are processed in accordance with priority in a multi-tasking environment which balances speed of execution among the different priority packets while making sure to complete all tasks. A variety of multi-tasking control techniques for completing multiple tasks simultaneously, giving priority to higher tasks, while devoting some system resources to complete the lowest priority tasks, are known to those skilled in the art.

Audio data playback is delayed in a delay 734, 742, as indicated above. Video data display is similarly delayed in delay 731, 739 to maintain synchronism between video and audio. The multi-task control 730, 738 sets the amount of delay (for both video and audio) in accordance with the predicted average delay of the communication channel 720, 746. Delayed audio is then displayed 736, 744 at the receiver for caller 2 and caller 3. Delayed video is simultaneously displayed 732, 740 at the receiver for caller 2 and caller 3. Furthermore, since callers 2 and 3 are

both receiving the same packets broadcast by caller 1, both hear and view the same multimedia messages.

Multimedia communication is typically two way between all callers. It should be understood that caller 1, caller 2 and caller 3 all include the transmitter and receiver elements shown in figures 7A and 7B. This does not mean, however, that all callers need to transmit or receive data. Each caller can choose to (or be required to) receive only or transmit only.

In operation, at caller 1 in figure 7A, successive multimedia data packets with assigned priority are generated 708, 710, 712 from respective multimedia sources 702, 704 and 706. As the packets are generated, each is placed 714 in priority order in a queue 716 and transmitted over a communication channel 718. If the channel capacity were unlimited, packets would be transmitted as soon as generated. However, in the normal case, generated packets may accumulate awaiting transmission because the communication channel capacity is limited. The present priority scheme assures that packets are transmitted in priority order with the highest priority packets transmitted first.

At the receiver, callers 2 and 3 in figure 7B both receive packets from the communication channel 720, 746. Received packets at callers 1 and 2, are processed in accordance with the received priority, to play back the received audio, video and display of the

received text/graphics. Since both callers 2 and 3 receive the same packets, a three party videoconference call is created.

Continuity of audio playback is perceived as critical to a multimedia conference. Accordingly, audio packets, being assigned the highest priority, are sent as soon as possible, and at the receiver, are processed as soon as possible after receipt. Thus, audio packets tend to be delivered in the shortest time which the communication channel will allow.

The system of the present invention tolerates transmission errors inherent in a traditional packet switched system by discarding or retransmitting corrupted audio or video. For text files, the normal error detection and retransmission requests are used. Sound and video are distinguished from text or file data in that it is possible to tolerate some loss of sound and video and still maintain an acceptable quality. In the event of a detected error in the received audio or video packet, the receiver determines whether there is sufficient time to flag the error and request a retransmission, based on the predicted average delay time of the system. If there is not sufficient time, the corrupted packet is ignored. In such manner, network capacity is not wasted on retransmissions which will arrive too late and have to be discarded anyway. However, the lowest priority packets of text/graphics or computer file data which are not time dependent, are flagged for errors and retransmitted.

OBJECT ORIENTED CPacketStream STREAMING METHOD

Various types of streams are used to achieve multimedia communications. First, a connection stream provides the interface to the communication channel. Then, there is a media stream for each desired media. For example, there may be a video stream, an audio stream, a video and audio stream such as QuickTime, or a text/data/graphics stream representing files, graphic images of many types, or any other data required. The architecture is designed to support "drop in" streams for new kinds of collaborative data.

The block diagram of figure 8 illustrates the method and apparatus for sending and receiving data packets, also termed C PACKets. Each of the above methods and tasks is described in detail below, including pseudo code for realizing each of the methods and tasks on a general purpose digital computer. Although the preferred embodiment is described in terms of software operating in a Macintosh computer environment, it will be understood that the present multiple media communication system of the present invention may be implemented in hardware, such as in dedicated logic, microprogrammed bit slices, programmable arrays and the like.

CPacketStream 80 is a software component which is responsible for handling C PACKets to establish communication channels between

machines. Each CPacketStream 80 is composed of a set of routines (or methods) responsible to interact with CPackets. These methods are used in turn by a set of tasks running in each CPacketStream. The task types and methods (or routines) available for packet handling are summarized as follows and described in more detail below.

TASKS:

WriteTask	(prioritized multi-tasking of received CPackets)
ReadTask	(connection stream distributes received CPackets)
IdleTask	(send final packet and initial packet)
OpenTask	(open connection stream)

METHODS

DoQueue	(puts a Cpacket in the WriteQueue)
DoWrite	(generates actual output from packet data)
DoIdle	(idle between packets)
Write	(lookups destination and calls DoQueue)
WriteDone	(acknowledges receipt of packet)
WriteQueue	(A buffer containing CPackets in priority order)
ReadQueue	(A buffer containing CPackets in received order)

CPacketStream::WriteTask 94

The WriteTask 94 is responsible for distributing packets contained in the WriteQueue 96 in each CPacketStream 80. The priority of this task is at least as high as the packet it is currently handling. This task is in a loop currently scanning the WriteQueue 96, if the queue is empty then the task will sleep. The CPacketStream::DoQueue method will put a CPacket into the WriteQueue 96, and wake the WriteTask 94 up. Therefore, the WriteTask 94 will be the one displaying or playing the packets.

```
CPacketStream::WriteTask
  if a packet in WriteQueue
    call DoWrite for that packet to handle data
  end
end
```

CPacketStream::ReadTask 82

The ReadTask 84 is responsible for reading CPackets from a particular channel, and redistributing among CPacketStreams 80 in that machine. This type of task is only appropriate for a connection (media) CPacketStream 80. (In a way it is similar to the WriteTask 94, serving a WriteQueue 96, but in the reverse direction, and corresponding to receiving data packets in a ReadQueue)

```
CPacketStream::ReadTask
  if a new packet read
    write a new packet
  end
end
```


CPacketStream::IdleTask 82

The idle task 82 is responsible for generating and sending 'helo' (the initial packet) and 'kiss' (the final packet) CPackets. It is also responsible to execute idle time events in some particular streams. For example, a Communications Tool (from Apple Computer) needs to have an idle call every so often in order to handle a particular connection.

```
CPacketStream::IdleTask
  if initial packet not sent
    if initial packet not created
      create initial packet
    end
    sent initial packet
  end
  idle stream if necessary
  if stream should die
    if final packet not created
      create final packet
    end
    send final packet
  end
  if final packet sent and stream should die
    disable and delete the streams
  end
end
```

An OpenTask 88 is used when the connection is being first opened and the connection negotiated between callers. At that time, the contents of the table lookup 98, which defines media types and connection streams is determined. In addition to these tasks, there are several methods that are called by these tasks in order to communicate with the stream. These methods are:

CPacketStream::DoQueue 86

This is the method that is called in order to give a packet to a particular stream. Most streams will immediately write the packet to a WriteQueue 96, and activate the WriteTask 94 in order to handle that particular packet.

```
CPacketStream::DoQueue 86
- put packet into the write queue
  wakeup WriteTask
end
```

CPacketStream::DoWrite 92

The WriteTask 94 will call this routine to actually handle the packet's content. For a connection stream, this is the output routine of a CPacketStream 80 to the communication channel. For a video stream, this routine will decompress and display the video contained in a packet. For other media streams, the DoWrite 92 routine will carry out the appropriate process to get the data displayed, played or otherwise.

```
CPacketStream::DoWrite
  handle the packet's data
end
```

CPacketStream::DoIdle

This is the routine which can be used to idle the CPacketStream 80. Many streams can use this to do periodic tasks.

```
CPacketStream::DoIdle
  perform periodic task
end
```

```
CPacketStream::Write 90
```

This routine will look up in table 98 the destination address for each destination in the packet, and the call DoQueue 86 for each destination packet stream. Each DoQueue 86 can refuse the packet until a later time, and therefore the packet contains flags for each destination address such that a complete write will mark that address completely written. A packet therefore contains an active destination count (72 in figure 6).

```
CPacketStream::Write
  for all destination addresses in packet
    lookup destination stream in address table
    if alias entry and 'info' packet
      add writeAsRemoteInfo flag
    end
    if found
      call DoQueue method for destination stream
      if successful
        mark destination address write complete
      end
    else
      mark destination address write deleted
    end
  end
end
```

Data packet flow and handling through CPacketStream 80 is from calls to DoQueue 86 which will write a CPacket into the WriteQueue 96, and then activate WriteTask 94, which processes the Cpackets, and calls DoWrite 92 to distribute the Cpackets contained in the WriteQueue 96.

CPacketStream::WriteDone

This routine will be called to dispose of the packet generated by a certain stream. It can be used for synchronization. A connection stream has the option of calling WriteDone to transfer acknowledge packets on the outgoing connection. The CPacketStream owns the packet which it sends, until all other streams are done with the packet. At that time, the packet ready to be deleted. However, when a packet (e.g., video) is sent from one machine on to another machine, such as between an ethernet LAN (local area network) and a telephone modem, the packet (e.g., the video) is not actually being used. In such circumstances, the originating connection stream should hold the packet, until all other connections have used this packet on the other machine(s). Synchronization of packet receipt is accomplished by returning an acknowledge packet when the WriteDone function of the connection stream is called at each machine which receives the packet. This is an additional form of communications between machines to reinforce the normal packet communications. "Acknowledge" packets have the same priority as the information packets, the lowest

packet priority.

STREAMING ALGORITHM

A generalized representation of the use of the present streaming algorithm is shown in the block diagram of figure 9. Two CPacketStreams, CPacketStream A, 801 and CPacketStream B, 802 are shown. By way of example, if CPacketStream A was a connection stream, then CPacketStream B would be a media stream, such as a video stream. On the other hand if CPacketStream A was a video stream, such as from a frame grabber, then CPacketStream B would be a connection stream. In general, there is one stream for each type of media, plus one stream for each connection. That is, a separate connection stream is used for each caller. Thus, for a two way conference with one other caller, there is one connection stream, while for a three way conference there are two connection streams, one for each of the other two callers. In an alternate embodiment, such as may be used with future higher speed communication systems, a single connection stream may be used with more than one caller.

Figure 9 also shows a lookup table 818 which is filled in when each stream is established for keeping track of the destination of the various packets. In operation, a packet is generated 804 and the Write function 806 is called. The pseudo code 816 for the Write function 806 contains a reference to a lookup to table 818, which returns an address to CPacketStream B, 802. CPacketStream B,

802 calls DoQueue 812, which writes the CPacket to WriteQueue 810. WriteTask 814 is activated to process the CPacket, which calls DoWrite 808 to generate the output routine of a CPacketStream 80 to the communication channel, or other appropriate media output.

Figure 10 illustrates the use of lookup tables to generate destination addresses from connection information between two given callers. By way of example, assume that CPacketStream A, 904 is a video stream connected to frame grabber 902 and image decompressor 912 at machine 1. A machine 2, CPacketStream D, 910 is also a video stream connected to a frame grabber 920 and image decompressor 918 at machine 2. Then, CPacketStream B, 906 is a connection stream coupled to a communication channel interface 914, such as for example a transmission source for an AppleTalk Data Streaming Protocol (ADSP) device. CPacketStream C, 908 is a connection stream coupled to a communication channel interface 916, shown as the receiving side of an AppleTalk Data Streaming Protocol device. Machine 1 uses table 922 to lookup the destination stream address 2,D for packets generated using data from video grabber 902. Similarly, machine 2 uses lookup table 925 to lookup the destination stream address 1,A for packets generated using data from video grabber 920.

PACKET ACKNOWLEDGEMENT

A block diagram illustrating the use of an optional

acknowledgement packet is shown in figure 11. A media stream 928, responsive to a video packet 936, calls the Write function, which through the appropriate lookup table, calls the DoQueue and DoWrite functions of connection stream 930, an ethernet connection stream. The video packet is transmitted on the ethernet communication channel 938 and received by the ReadTask and Write functions of connection stream 932. Thereafter, the DoQueue and DoWrite functions of media stream 934 are called through the appropriate lookup table which displays the data on video display 940.

The communication channel LAN protocol typically supports lower level acknowledgment functions. For example, it is known by the transmitting caller that the packet was received over the clear communication channel 938. Otherwise, the LAN protocol (at the ADSP level for example) would have returned an error indication. In addition to the acknowledge at the LAN protocol level, an acknowledge packet is generated when the received data is played (i.e., when the video data is displayed) in order to provide end to end synchronization information. The WriteDone function of connection stream 932 generates such acknowledge packet for return transmission across communication channel 938. Back at the originating transmitting caller, the ReadTask function of connection stream 930, calls WriteDone routine of media stream 928 to process the acknowledge packet. The receipt of an acknowledge packet also provides an indication of system delay for the media type of media stream 928, in this example, a video packet. The

acknowledge packet contains a recorded time indicating when the video packet was actually used. Comparison of the recorded transmission time with the received recorded display time, provides a measure of the end to end system delay.

PRIORITIZED DATA PACKET PROCESSING

A system block diagram is illustrated in figures 12 and 13. Figure 12 shows the transmission elements in solid lines and the receive elements in dotted lines. Figure 13 shows the receive elements in solid lines and the transmit elements in dotted lines.

In figure 12, audio data source 110 and video data source 112 are coupled through audio/video stream 114 and connection stream 116 to data communication channel 118. In figure 13, data communication channel 118 is coupled to connection stream 216, and then to audio/video stream 214. Audio is played by sound manager 220, which includes a FIFO delay buffer 228. Video is played back by video decompressor 224 coupled to video display device 226.

For the return direction, figure 13 also shows audio data source 210 and video data source 212, coupled to the communication channel 218 through audio/video stream 214 and connection stream 216. At the transmission side in figure 12, audio is played by sound manager 120, which includes a FIFO delay buffer 128. Video is played back by video decompressor 124 coupled to video display

device 126.

In operation in figure 12, data generated by audio source 110 and video data source 112 are placed into packets in audio/video stream 114, and forwarded to connection stream 116. The connection stream arranges the available packets in priority order before transmission on the network communication channel 118. Once transmission of a packet has begun, however, it is typically not interruptable. For example, if a video packet represents a video frame update, and the video packet transmission has started, no audio packet can be sent until the current packet is completed. If it is desired to improve audio transfer, the video frame update may be divided into smaller sub-frame packets. Then, an audio packet will be inserted during transmission of a complete video frame update, i.e., by insertion between sub-frame packets forming the video frame update.

In figure 13, data packets received by connection stream 216 are distributed to the audio/video stream 214. Audio data packets, having a higher priority represent a higher priority task. Thus, the sound manager 222 is given priority over the video decompressor 224. As indicated above, delay buffer 228 is set equal to the predicted average packet transmission delay of the communication system. Alternatively, the delay provided by delay buffer 228 is dynamically adjustable according to system delay as measured by time reported by return message packets or acknowledge packets.

Audio playback is slowed or accelerated in order to shrink or grow the difference in time between the sender and receiver.

Additional media types, such as file text or screen documents may be added to the block diagrams of figures 12 and 13 as additional inputs to the Write function of stream 114 in figure 12 and additional outputs of stream 214 in figure 13. In sharing collaborative documents, one member of the conference may be running the application such as a word processor or spreadsheet, and the others viewing a screen only. Alternatively, one member of the conference may be running the application, but the keystrokes of the others are transmitted back to the one member as text data. In such manner, conference members may each have direct input into the collaborative application.

As indicated, the preferred embodiment of the present invention is in software running on a Macintosh computer. A software embodiment has the advantage of being hardware independent, capable of working with any available media source, and across any available communication system. In addition, CPacketStream tasks and methods are shared by various connection streams and media streams. It is noteworthy that the present system achieves multimedia conferencing in a uniprocessor architecture.

Alternative embodiments of the present multimedia

communication system include multi-processor architectures where the multi-tasking of received multimedia data packets may be replaced by parallel processing, or in special purpose hardware. In dedicated hardware, each CPacketStream could be a special purpose microprogrammed integrated circuit, where one chip would be required for each media type, and for each connection.

Figures 14 through 18 illustrate the various capabilities of the present system of multiple media digital communication. Figure 14 illustrates a standard video call of the type shown in figure 2 over an ethernet network of the type illustrated. Figure 15 illustrates a video call with collaborative data over an ethernet network of the type illustrated on the screen in figure 3. This configuration is contemplated as the most common type of multimedia call.

Figure 16 illustrates a one way video/audio call with collaborative data over an ethernet network. The data is one way because first party did not answer, but that party was configured to accept messages. The received data is recorded in memory or on disk and played back later, thus creating a multimedia message answering machine. In the message record mode, system delays are not limiting because the message does not have to be recorded in real time; the only requirement is to play it back in real time. The message is recorded on one machine, and sent as a complete message file to the other machine, and there stored on the drive.